

OE-SS API Reference

v1.1.6

API Overview

OE-SS provides a web-services based API for programmatic interaction. This API provides 3 functional modules: general interrogation of network data, provisioning and measurement data. Within each module there are several methods. Using the API one can for instance provision a new VLAN across multiple switches, identify what resources are available on the network, or observe the amount of traffic traversing a provisioned vlan.

Protocol Description

Format of requests

All services are implemented using a JSON-RPC architecture. Requests to the services are sent in the HTTP Header. All of the services allow both POST and GET mechanisms for requests. The parameter name should be specified in the header followed by an '=' sign and then the value for that parameter. In perl all requests can be made through the LWP::UserAgent module, however any code that can access HTTP content can be used to access the services.

Format of responses

All services respond with JSON objects representing the requested results. All successful results are returned in side of a "results" object. Errors will be reported outside of the results object in an "error" object. Using perl all JSON objects can be parsed with the JSON module, and for improved speed use the JSON::XS module.

How to get help

Help using this API can be found at

For more information on OE-SS or to get the latest code please go to

<http://github.com/GlobalNOC/OESS>

The Software page can be found at

<http://globalnoc.iu.edu/sdn/oess.html>

and additional help can be found by emailing/joining the mailing list oess-users@grnoc.iu.edu

Authentication

Every OE-SS instance can be protected by a different authentication mechanism. By default OE-SS is protected by Apache's **Basic Auth** authentication and authorization service.

However each instance may use its own authentication mechanism. You should contact the administrator of the system to determine what type of Auth is best to use the Programmatic API. Using perl the authentication can be achieved by using the LWP::UserAgent module,

and specifically the credentials method. Instantiating an LWP::UserAgent and then specifying the credentials in the credentials method allows perl scripts to authenticate to the webservices.

Workgroups

Every user that authenticates to OEES should be a member of at least one workgroup. It is possible for a user to be in N number of workgroups, at which case they have to decide which group they are working for at the time. Each workgroup has a different set of permissions allowing where they can terminate endpoints. Once a user has logged in to OEES they are presented with the workgroup selection, this should also be the first step anyone using the API should do. The workgroup id is passed on to several other requests and allows OEES to determine which pieces of data the user is allowed to see.

Network Resource Service

Location: <https://<yourhost>/oess/services/data.cgi>

Description: Fetching general information about the state of the network, such as the network map, interfaces, workgroups, and circuits can be done through the General Data Fetching service (data.cgi). The results of this service are based on the user and the workgroup.

Method: get_node_interfaces

Description: returns a list of interfaces on the given node

Parameter	Value	Description
node	String	Name of the node to get a list of available interfaces for. This is dependent on the workgroup the user is participating in at the time
workgroup_id	Integer	The id of the workgroup the user is currently participating in

show_down (optional)	Boolean (1 0)	Show down interfaces on the node.
show_trunk (optional)	Boolean(1 0)	Show trunk interfaces on the node

Example:

```
action: get_node_interfaces
node: MLX_1
workgroup_id: 1
```

Results: {"results": [

```
    {
        "interface_id": "1262",
        "name": "e15/6",
        "description": "e15/6",
        "int_role": "unknown",
        "status": "up",
        "port_number": "674",
        "vlan_tag_range": "-1,1-4095"
    }
]
```

Method: get_shortest_path

Description: returns the shortest contiguous path between the given nodes

Parameter	Value	Description
node	String (array)	An array of node names to connect together with the shortest path
link	String (array)	A list of links to avoid when doing the shortest path calculation

Example:

```
action: get_shortest_path
node: nddi-sw.chic.net.internet2.edu
node: nddi-sw.rale.net.internet2.edu
```

Results: {"results": [

```
    {
        "link": "CHIC-NEWY-10GE"
    },
]
```

```

        {
          "link": "NEWY-WASH-10GE"
        },
        {
          "link": "WASH-RALE-10GE"
        }
      ],
    }
  }
}

```

Method: get_existing_circuits

Description: returns a list of circuits for the given workgroup

Parameter	Value	Description
workgroup_id	Integer	The workgroup ID to retrieve a list of circuits for. (Every circuit is associated to a workgroup)
path_node_id (optional)	Integer	Filters the results for circuits that traverse the node of the node_id given
endpoint_node_id (optional)	Integer	Filters the results to circuits that terminate on the specified node_id

Example:

```

action: get_existing_circuits
workgroup_id: 1

```

Results: {"results": [

```

      {
        "circuit_id": "111",
        "bandwidth": "750",
        "name": "GRNOC-dd3ddfd2-eaf5-11e1-b638-782bcb498498",
        "description": "Test Circuit",
        "endpoints": [
          {
            "interface": "63",
            "node": "nddi-sw.newy32aoa.net.internet2.edu"
          },
          {
            "interface": "63",
            "node": "nddi-sw.chic.net.internet2.edu"
          }
        ],
        "links": [
          {
            "interface_z": "e1/2",
            "port_no_z": "2",
            "node": "nddi-sw.chic.net.internet2.edu"
          }
        ]
      }
    ]
  }
}
```

```

        node_z: "sdn-sw.denv.net.internet2.edu",
        port_no_a: "1",
        node_a: "sdn-sw.kans.net.internet2.edu",
        name: "KANS-DENV-100GE",
        interface_a: "e1/1"
    },
],
"state": "active",
active_path: "backup",
bandwidth: "0",
internal_ids:
{
    primary:
    {
        sdn-sw.denv.net.internet2.edu: "2",
        sdn-sw.kans.net.internet2.edu: "2"
    }
},
last_edited: "9/28/2012 18:0:18",
user_id: "1",
operational_state: "up",
created_by:
{
    email: "aragusa@grnoc.iu.edu",
    is_admin: "0",
    auth_id: "3381",
    given_names: "Andrew",
    user_id: "11",
    family_name: "Ragusa",
    auth_name: "aragusa"
}
}
]
}

```

Method: get_circuit_details

Description: returns all of the details for a given circuit

Parameter	Value	Description
circuit_id	Integer	The id of the circuit to fetch details for

Example:

```

action: get_circuit_details
circuit_id: 111

```

Results: {"results":

```

{
    "circuit_id": "111",
    "name": "GRNOC-dd3ddfd2-eaf5-11e1-b638-782bcb498498",
    "external_identifier": null,
    "description": "Test Circuit",
    "endpoints": [
        {
            "local": "1",
            "node": "nddi-sw.newy32aoa.net.internet2.edu",
            "port_no": "63",
            "node_id": "12",
            "urn": "urn:ogf:network:domain=some.net:node=somehost:port=63:link=*",
            "interface": "63",
            "tag": "500",
            "role": "unknown"
        }
    ]
}
```

```

        },
        {
            "local": "1",
            "node": "nddi-sw.chic.net.internet2.edu",
            "port_no": "63",
            "node_id": "41",
            "urn": "urn:ogf:network:domain=some.net:node=somehost:port=63:link=*",
            "interface": "63",
            "tag": "500",
            "role": "unknown"
        }
    ],
    "state": "active",
    "backup_links": [],
    "bandwidth": "750",
    "active_path": "primary",
    "internal_ids": {
        "primary": {
            "nddi-sw.chic.net.internet2.edu": "51",
            "nddi-sw.newy32aoa.net.internet2.edu": "50"
        }
    },
    "links": [
        {
            "interface_z": "17",
            "port_no_z": "17",
            "node_z": "nddi-sw.chic.net.internet2.edu",
            "port_no_a": "18",
            "node_a": "nddi-sw.newy32aoa.net.internet2.edu",
            "name": "CHIC-NEWY-10GE", "interface_a": "18"
        }
    ],
    "workgroup": {
        workgroup_id: "11",
        external_id: null,
        status: "active",
        name: "GRNOC",
        type: "admin",
        description: ""
    },
    active_path: "backup",
    bandwidth: "0",
    internal_ids: {
        primary: {
            {
                sdn-sw.denv.net.internet2.edu: "2",
                sdn-sw.kans.net.internet2.edu: "2"
            }
        },
        last_edited: "9/28/2012 18:0:18",
        user_id: "1",
        operational_state: "up",
        created_by: {
            email: "aragusa@grnoc.iu.edu",
            is_admin: "0",
            auth_id: "3381",
            given_names: "Andrew",
            type: "normal",
            user_id: "11",
            family_name: "Ragusa",
            auth_name: "aragusa"
        }
    }
}

```

Method: get_circuit_details_by_external_identifier

Description: Returns the same data as get_circuit_details but finds the circuit based on some external id that was passed to the system at creation or last modification.

Parameter	Value	Description
external_identifier	String	Whatever was used as the external identifier when the circuit was created or last modified.

Example:

```
action: get_circuit_details_by_external_identifier
external_identifier: foo
```

Results: {"results":

```
{
    "circuit_id": "111",
    "name": "GRNOC-dd3ddfd2-eaf5-11e1-b638-782bcb498498",
    "description": "Test Circuit",
    "endpoints": [
        {
            "local": "1",
            "node": "nddi-sw.newy32aoa.net.internet2.edu",
            "port_no": "63",
            "node_id": "12",
            "urn": "urn:ogf:network:domain=some.net:node=somehost:port =63:link=*",
            "interface": "63",
            "tag": "500",
            "role": "unknown"
        },
        {
            "local": "1",
            "node": "nddi-sw.chic.net.internet2.edu",
            "port_no": "63",
            "node_id": "41",
            "urn": "urn:ogf:network:domain=some.net:node=somehost:port=63:link=*",
            "interface": "63",
            "tag": "500",
            "role": "unknown"
        }
    ],
    "state": "active",
    "backup_links": [],
    "bandwidth": "750",
    "active_path": "primary",
    "internal_ids": {
        "primary": {
            "nddi-sw.chic.net.internet2.edu": "51",
            "nddi-sw.newy32aoa.net.internet2.edu": "50"
        }
    },
    "links": [
        {
            "interface_z": "17",
            "port_no_z": "17",
            "node_z": "nddi-sw.chic.net.internet2.edu",
            "port_no_a": "18",
            "node_a": "nddi-sw.newy32aoa.net.internet2.edu",
            "name": "CHIC-NEWY-10GE", "interface_a": "18"
        }
    ]
}
```

```

        ],
        "workgroup": {
            "workgroup_id": "11",
            "external_id": null,
            "name": "GRNOC",
            "type": "admin",
            "description": ""
        },
        "active_path": "backup",
        "bandwidth": "0",
        "internal_ids": {
            "primary": {
                "sdn-sw.denv.net.internet2.edu": "2",
                "sdn-sw.kans.net.internet2.edu": "2"
            }
        },
        "last_edited": "9/28/2012 18:0:18",
        "user_id": "1",
        "operational_state": "up",
        "created_by": {
            "email": "aragusa@grnoc.iu.edu",
            "is_admin": "0",
            "auth_id": "3381",
            "given_names": "Andrew",
            "type": "normal",
            "user_id": "11",
            "family_name": "Ragusa",
            "auth_name": "aragusa"
        }
    }
}

```

Method: get_circuit_scheduled_events

Description: returns a list of scheduled circuit events. These events can be anything from creation, modification or removal from the network.

Parameter	Value	Description
circuit_id	Integer	The id of the circuit fetch scheduled events for

Example:

```

action: get_circuit_scheduled_events
circuit_id: 291

```

Results:{ results:

```

    [
        {
            activated: "2013-08-01 00:00:00",
            scheduled_action_id: "81",
            scheduled: "2013-05-16 17:54:09",
            username: "aragusa",
            layout: "<opt name="" action="remove" version="1.0" /> ",
            fullname: "Andrew Ragusa",
            user_id: "11",
            completed: null
        }
    ]
}

```

Method: get_circuit_history

Description: returns a list of network events that have affected this circuit, including port status changes, link failures and failovers, and node outages

Parameter	Value	Description
circuit_id	Integer	The id of the circuit fetch network events for

Example:

```
action: get_circuit_history
circuit_id: 291
```

Results :{results:

```
[{
    layout: "",
    activated: "2013-05-14 14:28:53",
    fullname: "Andrew Ragusa",
    scheduled: -1,
    completed: null,
    username: "aragusa"
},
{
    layout: "",
    activated: "2013-05-13 17:42:12",
    fullname: "Andrew Ragusa",
    scheduled: -1,
    completed: "2013-05-13 17:42:12",
    username: "aragusa"
}]
}
```

Method: is_vlan_tag_available

Description: returns the availability of the vlan tag for a given node and interface

Parameter	Value	Description
interface	String	The name of the interface to check the vlan tags availability
node	String	The name of the node the interface is on
vlan_tag	Integer	The vlan tag to check the availability of on the node/interface combination

Example:

```
action: is_vlan_tag_available
vlan: 100
interface: 63
node: nddi-sw.chic.net.internet2.edu
```

Results: {"results":

```

        [
            {"available":1}
        ]
    }
}
```

Method: get_maps

Description: returns a JSON object representing the network layout. This is used by the frontend for drawing the network diagrams in OpenLayers.

No Parameters accepted

Example:

action: get_maps

```

results: {"results": [
    {
        "nodes": {
            "HOST_2": {
                "node_long": "-102",
                "vlan_range": "2- 4095",
                "node_id": "41",
                "node_name": "MLX_2",
                "node_lat": "42"
            },
            "HOST_1": {
                "node_long": "-92",
                "vlan_range": "2-4095",
                "node_id": "40",
                "node_name": "HOST_1",
                "node_lat": "35"
            }
        },
        "links": {
            "HOST_2": [
                {
                    "link_id": "34",
                    "to": "MLX_1",
                    "link_state": "up",
                    "capacity": "10000",
                    "link_name": "100GE",
                    "remote_urn": null
                }
            ],
            "HOST_1": [
                {
                    "link_id": "34",
                    "to": "HOST_2",
                    "link_state": "up",
                    "capacity": "10000",
                    "link_name": "100GE",
                    "remote_urn": "
                }
            ]
        }
    }
]}
```

urn:ogf:network:domain=ion.internet2.edu:node=rtr.seat:port=et-5/0/0:link=al2s"

```

        {
            "network_name": "somedomain.net",
            "network_long": "0",
            "local": 1,
            "network_lat": "0"
        }
    ]
}

```

Method: get_workgroups

Results:

Description: returns a list of workgroups the logged in user has access to.

No Parameters accepted

Example:

```
action:get_workgroups
```

Results: {"results": [

```

        {
            "workgroup_id": "1",
            "name": "test"
        }
    ]
}
```

Method: generate_clr

Results:

Description: generates a human readable Circuit Layout Record describing the given circuit. If the raw option is specified the results contain the flow rules on the devices.

Parameter	Value	Description
circuit_id	Integer	The circuit_id of the circuit to have the CLR generated for
raw	Boolean (1 0)	Generate flow rule view of the circuit for every node

Example:

```
action:generate_clr
circuit_id: 1
```

Results: {"results":

```

        {
            "clr": "Circuit test-71564d06-bbf4-11e2-88bd-782bcb48ff73 Created by Andrew Ragusa at
            5/13/2013 17:42:12 for workgroup test Lasted Modified By: Andrew Ragusa at 5/14/2013 14:28:53
            Endpoints: brocade-1.sdn-test.grnoc.iu.edu - e1/1 VLAN 500 brocade-2.sdn-test.grnoc.iu.edu - e15/2
            VLAN 500 Primary Path: 10GE2 Backup Path: 100GE 100GE2 10GE "
        }
    }
```

Method: get_all_node_status

Results:

Description: returns a list of all active nodes and their operational status.

No Parameters accepted

Example:

```
action:get_all_node_status
```

Results: {results:

```
[ {  
    node_id: "11",  
    dpid: "155568786688",  
    name: "foo.testlab.net",  
    operational_state: "up"  
,  
{  
    node_id: "21",  
    dpid: "155569112064",  
    name: "bar.testlab.net",  
    operational_state: "up"  
}  
]
```

Method: get_all_link_status

Results:

Description: returns a list of all active links and their current operational status

No Parameters accepted

Example:

```
action:get_all_link_status
```

Results: {results:

```
[ {  
  
    fv_status: "up",  
    status: "up",  
    remote_urn: null,  
    start_epoch: "1368134958",  
    metric: "1",  
    name: "10GE2",  
    link_id: "1",  
    end_epoch: "-1",  
    link_state: "active",  
    interface_z_id: "51",  
    interface_a_id: "91"  
,  
{  
    fv_status: "up",  
    status: "up",  
    remote_urn: null,  
    start_epoch: "1368134961",  
    name: "100GE",  
    metric: "1",  
    link_id: "11",  
    end_epoch: "-1",  
    link_state: "active",  
    interface_z_id: "121",  
}
```

```

        interface_a_id: "171"
    },
{
    fv_status: "up",
    status: "up",
    remote_urn: null,
    start_epoch: "1368134962",
    name: "100GE2",
    link_id: "21",
    end_epoch: "-1",
    metric: "1",
    link_state: "active",
    interface_z_id: "41",
    interface_a_id: "131"
}
]
}

```

Method: get_all_resources_for_workgroup

Results:

Description: returns a list of all resources (endpoints) for which the workgroup has access. *NOTE* the current user must be a part of this workgroup to get any results.

Parameter	Value	Description
workgroup_id	Integer	The workgroup_id of the workgroup to get a list of the resources (and their status) for.

Example:

```
action:get_all_resources_for_workgroup
workgroup_id: 1
```

Results: {results:

```

    [
        {
            interface_name: "e15/2",
            node_name: "foo.testlab.net",
            node_id: "11",
            remote_links: [],
            owning_workgroup: {
                workgroup_id: "2",
                status: "active",
                name: "mininet",
                max_circuit_endpoints: "10",
                description: "",
                max_circuits: "20",
                external_id: undef,
                type: "normal",
                max_mac_address_per_end: "10"
            },
            interface_id: "71",
            description: "e15/2",
            operational_state: "down"
        },
        {
            interface_name: "e15/2",
            node_name: "bar.testlab.net",
            node_id: "21",
            owning_workgroup: {
                workgroup_id: "2",

```

```

        status: "active",
        name: "mininet",
        max_circuit_endpoints: "10",
        description: "",
        max_circuits: "20",
        external_id: undef,
        type: "normal",
        max_mac_address_per_end: "10"
    },
    interface_id: "111",
    remote_links: [],
    description: "e15/2",
    operational_state: "up"
},
{
    interface_name: "xe-7/0/0.0",
    node_name: "foobar.testlab.net",
    node_id: "41",
    remote_links: [],
    owning_workgroup: {
        workgroup_id: "2",
        status: "active",
        name: "mininet",
        max_circuit_endpoints: "10",
        description: "",
        max_circuits: "20",
        external_id: undef,
        type: "normal",
        max_mac_address_per_end: "10"
    },
    interface_id: "191",
    description: "xe-7/0/0.0",
    operational_state: "down"
},
{
    interface_name: "xe-7/0/0.0",
    node_name: "wha.testlab.net",
    node_id: "31",
    remote_links: [],
    owning_workgroup: {
        workgroup_id: "2",
        status: "active",
        name: "mininet",
        max_circuit_endpoints: "10",
        description: "",
        max_circuits: "20",
        external_id: undef,
        type: "normal",
        max_mac_address_per_end: "10"
    },
    owned_by:{
        interface_id: "151",
        description: "xe-7/0/0.0",
        operational_state: "up"
    }
}

```

Method: get_workgroup_members

Results:

Description: returns a list of users in the specified workgroup. *NOTE* The current user must be in the workgroup to have any results returned.

Parameter	Value	Description
workgroup_id	Integer	The workgroup_id of the workgroup that should have its users returned

Example:

```
action:get_workgroup_members
workgroup_id: 1
```

Results: {results:

```
  [
    {
      email_address: "aragusa@grnoc.iu.edu",
      user_id: "11",
      family_name: "Ragusa",
      auth_name: [
        "aragusa"
      ],
      first_name: "Andrew"
    },
    {
      email_address: "gmcnaugh@grnoc.iu.edu",
      user_id: "22",
      family_name: "McNaught",
      auth_name: [
        "gmcnaugh"
      ],
      first_name: "Grant"
    }
  ]
```

Method: get_circuits_by_interface_id

Description: returns a list of circuits on an interface

Parameter	Value	Description
interface_id	Integer	The circuit ID to be removed from the network

Example:

```
action: get_circuits_by_interface_id
interface_id: 1
workgroup_id: 1
```

Results: {"results":

```
  [
    {"circuit_id":1,
     "name": "Foo",
     "description": "this is a test circuit"
    },
    {"circuit_id": 2,
     "name": "Foo2",
     "description": "this is another test circuit"
    }
  ]}
```

Provisioning Service

Location: <https://<yourhost>/oess/services/provisioning.cgi>

The provisioning data service allows users to request circuit additions and removals from the network.

Method: provision_circuit

Description: Adds or modifies a circuit on the network. If circuit_id is undefined or -1, then the circuit is added. If a circuit_id is specified then the circuit is modified. It then returns a JSON object containing either a success message or an error message, indicating the circuit has been added successfully added or scheduled for addition/modification to the network.

Parameter	Value	Description
workgroup_id	Integer	The workgroup_id with permission to build the circuit, the user must be a member of this workgroup
external_identifier	Integer	
circuit_id	Integer	-1 or undefined indicated circuit is to be added. Any other value indicates a circuit Modification is to take place
description	String	A user specified description of the purpose of the circuit
bandwidth	Integer	The dedicated bandwidth of the circuit in Mbps (does nothing today)
provision_time	Integer	Timestamp of when circuit should be created in epoch time format. -1 means now
remove_time	Integer	The time the circuit should be removed from the network in epoch time format. -1 means never
link	String (array)	Array of names of links to be used in the primary path
backup_link	String (array)	Array of names of links to be used as backup paths
node	String (array)	Array of Nodes to be used
interface	String (array)	Array of interfaces to be used. Note that interface[0] is on node[0]

		and interface[1] is on node[1] etc...
tag	Integer (array)	An array of vlan tags to be used on each interface. Note that tag[0] is on interface[0] and tag[1] is on interface[1]
remote_nodes (unused)	URN (array)	Array of OSCARS URNs to use as endpoints for IDC based circuits
remote_tags (unused)	Integer (array)	VLAN tags to be used on the IDC endpoints
restore_to_primary (optional)	Integer	Time in minutes to restore to primary (setting to 0 disables restore to primary)

Example:

```

action: provision_circuit
circuit_id: -1
description: foo
bandwidth: 0
provision_time: -1
remove_time: -1
workgroup_id: 1
node: nddi-sw.chic.net.internet2.edu
interface: 63
tag: 100
node: nddi-sw.wash2.net.internet2.edu
interface: 63
tag: 100
link: CHIC-NEWY-10GE
link: NEWY-WASH-10GE
backup_link: SEAT-CHIC-10GE
backup_link: LOSA-SEAT-10GE
backup_link: LOSA-RALE-10GE
backup_link: WASH-RALE-10GE

```

Results: {"results":

```

    {
      "circuit_id": "<circuit_id>",
      "success": 1
    }
  }

```

Method: remove_circuit

Description: Removes a circuit from the network, and returns success if the circuit has been removed successfully or scheduled for removal from the network.

Parameter	Value	Description
-----------	-------	-------------

circuit_id	Integer	The circuit ID to be removed from the network
remove_time	Integer	The time for the circuit to be removed in epoch time. -1 means now
workgroup_id	Integer	the workgroup_id to remove the circuit as

Example:

```
action: remove_circuit
circuit_id: 131
remove_time: -1
workgroup_id: 1
```

Results: { "results":

```
[  
    {"success":1}  
]
```

```
}
```

Method: fail_over_circuit

Description: Changes a circuit over to its backup path (if it has one)

Parameter	Value	Description
circuit_id	Integer	The circuit ID to be removed from the network
workgroup_id	Integer	The time for the circuit to be removed in epoch time. -1 means now

Example:

```
action: fail_over_circuit
circuit_id: 131
workgroup_id: 1
```

Results: { "results":

```
[  
    {"success":1}  
]
```

```
}
```

Method: reprovision_circuit

Description: Removes and re-installs all flow rules related to a circuit on the network (useful for trouble shooting a circuit)

Parameter	Value	Description
circuit_id	Integer	The circuit ID to be removed from the

		network
workgroup_id	Integer	The time for the circuit to be removed in epoch time. -1 means now

Example:

```
action: reprovision_circuit
circuit_id: 131
workgroup_id: 1
```

Results:

```
{"results": [
    [
        {"success":1}
    ]
}
```

Measurement

Location: <https://<yourhost>/oess/services/measurement.cgi>

The measurement API provides measurement data used in the OE-SS UI for graphing purposes.

Method: get_circuit_data

Description: returns JSON formatted usage statistics for a circuit from start time to end time, and for a specific node or interface in the circuit.

Parameter	Value	Description
start	Integer	Start time in epoch seconds
end	Integer	End time in epoch seconds
circuit_id	Integer	The ID of the circuit to fetch statistics for
node	String	Optional parameter specifying the node to look at for the circuit usage data
interface	String	The name of the interface on the node to look at for the usage statistics. Must be specified when the node parameter is defined
link	String	Name of the link to view data for, if specified node/interface should not be specified

Monitoring

Location: <https://<yourhost>/oess/services/monitoring.cgi>

The measurement API provides monitoring data for external services to check on the status of the network.

Method: get_node_status

Description: returns JSON formatted status updates related to a Nodes connection state to the controller.

Parameter	Value	Description
node	String	Name of the node to query the status.

Example:

```
action: get_node_status
node: foo.testlab.net
```

Results: {"results": {"status": 1, "node": "foo.testlab.net"}}

Method: get_rules_on_node

Description: returns the maximum allowed rules on a switch, and the total number of rules currently on the switch.

Parameter	Value	Description
node	String	Name of the node to query the status.

Example:

```
action: get_rules_on_node
node: foo.testlab.net
```

Results: {"results": {"maximum_allowed_rules_on_switch": 4000, "rules_currently_on_switch": 15, "node": "foo.testlab.net"} }

Traceroute

Location: <https://<yourhost>/oess/services/traceroute.cgi>

Provides the functionality to initiate a traceroute and retrieve the results.

Method: init_circuit_traceroute

Description: starts a traceroute for a circuit

Parameter	Value	Description
-----------	-------	-------------

circuit_id	Integer	The identifier for the circuit in the OESS database
workgroup_id	Integer	the workgroup requesting the trace
node	String	the name of the node to start the trace from
interface	String	the name of the interface to start the trace from

Example:

```
action: init_circuit_traceroute
node: sdn-sw.chic.net.internet2.edu
interface: eth15/2
workgroup: 1
circuit_id: 1
```

Results: {"results": [{"success": 1}] }

Method: [get_circuit_traceroute](#)

Description: fetches the current results for a circuit traceroute

Parameter	Value	Description
circuit_id	Integer	The identifier for the circuit in the OESS database
workgroup_id	Integer	the workgroup requesting the trace

Example:

```
action: get_circuit_traceroute
workgroup: 1
circuit_id: 1
```

Results: {"results": [

```
  {"interfaces_traversed": ["eth1/2"],
   "end_epoch": "1430859451",
   "remaining_endpoints": "0",
   "nodes_traversed":
     ["brocade2.sdn-test.grnoc.iu.edu"],
   "ttl": "29",
   "status": "Complete",
   "start_epoch": "1430859435"}]
```

Remote Network / IDC

Location: <https://<yourhost>/oess/services/remote.cgi>

Provides a very simple provisioning API to access OSCARS topologies, circuit status, circuit provision, circuit modification, and circuit teardown.

Method: `get_networks`

Description: returns a JSON formatted output of the data returned from the OSCARS topology service specified in the OEES config. This will contain a list of all possible endpoints for Inter-domain Circuits.

Parameter	Value	Description
node	String	Name of the node to query the status.

Example:

```
action: get_networks
```

Results:

```
{results: [
  {
    urn: "urn:ogf:network:domain=al2s.net.internet2.edu",
    name: "al2s.net.internet2.edu",
    links: [
      {
        link: "ASHB-WASH-100GE",
        longitude: "-77.459206",
        urn: "urn:ogf:network:domain=al2s.net.internet2.edu:node=sdn-
sw.ashb.net.internet2.edu:port=et-5/0/0.0:link=ASHB-WASH-100GE",
        latitude: "39.016746",
        remote_urn: "urn:ogf:network:domain=al2s.net.internet2.edu:
node=sdn-sw.wash.net.internet2.edu:
port=e5/1:link=ASHB-WASH-100GE",
        port: "et-5/0/0.0",
        node: "sdn-sw.ashb.net.internet2.edu"
      },
      {
        link: "ASHB-CLEV-100GE",
        longitude: "-77.459206",
        urn: "urn:ogf:network:domain=al2s.net.internet2.edu:node=sdn-
sw.ashb.net.internet2.edu:port=et-7/0/0.0:link=ASHB-CLEV-100GE",
        latitude: "39.016746",
        remote_urn: "urn:ogf:network:domain=al2s.net.internet2.edu:
node=sdn-clev.net.internet2.edu:
port=e3/1:link=ASHB-CLEV-100GE",
        port: "et-7/0/0.0",
        node: "sdn-sw.ashb.net.internet2.edu"
      }
    ],
    urn: "urn:ogf:network:domain=ancpi.ro",
    name: "ancpi.ro",
    links: [
      {
        link: "*",
        longitude: null,
        urn: "urn:ogf:network:domain=ancpi.ro:node=unnamed-
10:port=dp1:link=*",
        latitude: null,
        remote_urn: "urn:ogf:network:domain=*:node=*:port=*:link=*",
        port: "dp1",
        node: "ancpi.ro-unnamed-10"
      }
    ]
  }
]}
```

```

        },
        {
            link: "*",
            longitude: null,
            urn: "urn:ogf:network:domain=ancpi.ro:node=unnamed-10:
                  port=s10-eth1:link=*",
            latitude: null,
            remote_urn: "urn:ogf:network:domain=:node=:port=:link=*",
            port: "s10-eth1",
            node: "ancpi.ro-unnamed-10"
        }
    ]
}

```

Method: create_reservation

Description: submits a circuit reservation to OSCARS and returns the GRI of the newly created reservation

Parameter	Value	Description
src_urn	String	Source URN as would be specified in the OSCARS UI. Must be in the topology server.
dst_urn	String	Destination URN as would be specified in the OSCARS UI. Must be in the topology server.
srv_vlan	Integer	Vlan id to use on the source side of the circuit
dst_vlan	Integer	Vlan id to use on the destination side of the circuit.
bandwidth	Integer	The bandwidth of the circuit in Mbps (must have a minimum of 50)
start_time	Integer (epoch time)	The start time (in epoch seconds) for the circuit to be created
end_time	Integer (epoch time)	The end time (in epoch seconds) for the circuit to be removed
description	String	A brief description of the purpose of the circuit

Example:

```

action: create_reservation
srv_urn: urn:ogf:network:domain=al2s.net.internet2.edu:node=sdn-sw.atla.net.internet2.edu:port=e15/2:link=*
dst_urn: urn:ogf:network:domain=al2s.net.internet2.edu:node=sdn-sw.wash.net.internet2.edu:port=e15/3:link=*
src_vlan: 500

```

```

dst_vlan: 500
bandwidth: 650
start_time: 1368737668
end_time: 1368737900

```

Results: {"results": [{"gri": "blah.foo.net-1", gti: 1}]}

Method: query_reservation

Description: Queries the current state of a circuit that is or has been provisioned through OSCARS. Either circuit_id or gri are required, however there is no reason to use both.

Parameter	Value	Description
circuit_id	Integer	OESS Circuit ID of the OSCARS circuit to query information for
gri	String	The OSCARS Global Reservation ID of the circuit in OSCARS

Example:

```

action: query_reservation
gri: al2s.net.internet2.edu-2191

```

Results: {"results":

```

[{
    status: "ACTIVE",
    path: [
        {
            from_lon: "-84.38759",
            to_node: "sdn-sw.atla.net.internet2.edu",
            from_node: "sdn-sw.atla.net.internet2.edu",
            to_lat: "33.758537",
            to_lon: "-84.38759",
            from_lat: "33.758537"
        },
        {
            from_lon: "-84.38759",
            to_node: "sdn-sw.chic.net.internet2.edu",
            from_node: "sdn-sw.atla.net.internet2.edu",
            to_lat: "41.896504",
            to_lon: "-87.64306",
            from_lat: "33.758537"
        },
        {
            from_lon: "-87.64306",
            to_node: "sdn-sw.chic.net.internet2.edu",
            from_node: "sdn-sw.chic.net.internet2.edu",
            to_lat: "41.896504",
            to_lon: "-87.64306",
            from_lat: "41.896504"
        }
    ],
    message: ""
}
]
```

Method: cancel_reservation

Description: cancels an OSCARS circuit reservation in whatever state the circuit is currently in.

Parameter	Value	Description
circuit_id	Integer	Circuit ID of the OESS circuit to be removed from OSCARS

Example:

```
action: cancel_reservation
circuit_id: 1
```

Results: {"results": [{"status": "IN_CANCEL", "message": "CANCELING CIRCUIT", "gri => 'al2s.net.internet2.edu-2191'}]}

Method: modify_reservation

Description: Modifies an existing OSCARS circuit reservation, and returns the GRI and GTI from OSCARS.

Parameter	Value	Description
circuit_id	Integer	the OESS circuit representing the OSCARS circuit
src_urn	String	Source URN as would be specified in the OSCARS UI. Must be in the topology server.
dst_urn	String	Destination URN as would be specified in the OSCARS UI. Must be in the topology server.
srv_vlan	Integer	Vlan id to use on the source side of the circuit
dst_vlan	Integer	Vlan id to use on the destination side of the circuit.
bandwidth	Integer	The bandwidth of the circuit in Mbps (must have a minimum of 50)
start_time	Integer (epoch time)	The start time (in epoch seconds) for the circuit to be created
end_time	Integer (epoch time)	The end time (in epoch seconds) for the circuit to be removed
description	String	A brief description of the

		purpose of the circuit
--	--	------------------------

Example:

```
action: modify_reservation
circuit_id: 1
srv_urn: urn:ogf:network:domain=al2s.net.internet2.edu:node=sdn-sw.atla.net.internet2.edu:port=e15/2:link=*
dst_urn: urn:ogf:network:domain=al2s.net.internet2.edu:node=sdn-sw.wash.net.internet2.edu:port=e15/3:link=*
src_vlan: 500
dst_vlan: 500
bandwidth: 650
start_time: 1368737668
end_time: 1368737900
```

Results: {"results": [{"gri": "al2s.net.internet2.edu-1111", "gti": 1}]}

Workgroup/ACL Management

Location: https://<yourhost>/oess/services/workgroup_manage.cgi

Provides the services to add/update/delete ACLs for a users workgroup

Method: get_all_workgroups

Description: returns a JSON formatted output of a list of all current workgroups in OESS.
No Parameters

Example:

```
action: get_all_workgroups
```

Results: {"results": [{"workgroup_id": "121", "external_id": null, "name": "12 AutoGOLE", "type": "normal"}, {"workgroup_id": "481", "external_id": null, "name": "AJ Net", "type": "normal"}, {"workgroup_id": "321", "external_id": null, "name": "AtlanticWave", "type": "normal"}]}

Method: get_acls

Description: Returns a JSON formatted list of ACLs for a given interface

Parameter	Value	Description
interface_id	Integer (Optional)	the interface_id to gather all ACLS for

interface_acl_id	Integer (Optional)	the specific interface_acl_id to request the details of
------------------	--------------------	---

Example:

```
action: get_acls
interface_id=1
```

Results: {results: [{

```
    interface_name: "eth1/1",
    vlan_end: null,
    vlan_start: "201",
    owner_workgroup_name: "Foo",
    workgroup_id: "1",
    interface_id: "1",
    interface_acl_id: "9231",
    workgroup_name: "Test1",
    eval_position: "1",
    owner_workgroup_id: "2",
    notes: "AN ACL!",
    allow_deny: "allow"
  },
  {
    interface_name: "eth3/2",
    vlan_end: "4089",
    vlan_start: "2",
    owner_workgroup_name: "Foo",
    workgroup_id: "1",
    interface_id: "1",
    interface_acl_id: "10021",
    workgroup_name: "Test2",
    eval_position: "1",
    owner_workgroup_id: "2",
    notes: "Another ACL",
    allow_deny: "allow"
  }
]}
```

Method: add_acl

Description: Adds an ACL for a specific interface/workgroup combination. A few things to note about how ACLs work; ACLs are evaluated in order, and short cut as soon as one matches. For example if there are 2 rules 1 that allows VLAN 100 and then another that denies all VLANs, and the allow rule is above the deny VLAN 100 will be allowed.

Parameter	Value	Description
workgroup_id	Integer	the workgroup the ACL is applied to
interface_id	Integer	Specific interface_id add the ACL to
allow_deny	String (allow deny)	if the ACL is an allow or deny rule
eval_position	Integer	the position in the ACL list where the rule will be evaluated
vlan_start	Integer	the start vlan tag

vlan_end	Integer	the end vlan tag
notes	String	Any notes or reason for the ACL (user optional)

Example:

```
action: add_acl
workgroup_id=1
interface_id=1
allow_deny=allow
eval_position=1
vlan_start=100
vlan_end=200
notes=A new ACL
```

Results: {results: [{success => 1,
 interface_acl_id => 100
 }]}{}

Method: update_acl

Description: updates an existing ACL. Allows for all ACL fields to be updated.

Parameter	Value	Description
interface_acl_id	Integer	the interface_acl_id of the ACL to be modified
workgroup_id	Integer	the workgroup the ACL is applied to
interface_id	Integer	Specific interface_id add the ACL to
allow_deny	String (allow deny)	if the ACL is an allow or deny rule
eval_position	Integer	the position in the ACL list where the rule will be evaluated
vlan_start	Integer	the start vlan tag
vlan_end	Integer	the end vlan tag
notes	String	Any notes or reason for the ACL (user optional)

Example:

```
action: update_acl
interface_acl_id=1
workgroup_id=1
interface_id=1
allow_deny=allow
eval_position=1
vlan_start=200
vlan_end=300
notes=Just update the ACL
```

Results: {results: [{success => 1}]}{}

Method: remove_acl

Description: removes an existing ACL

Parameter	Value	Description
interface_acl_id	Integer	the interface_acl_id of the ACL to be removed

Example:

```
action: delete_acl
interface_acl_id=1
```

Results: {results: [{ success => 1 }]}

Example interaction in perl

```
#!/usr/bin/perl

use strict;
use LWP::UserAgent;
use HTTP::Request;
use Data::Dumper;
use CGI;
use JSON::XS;

my $base_url = "https://<yourhost>/oess/";

my $workgroup;

my $query = new CGI("action=get_workgroups");
my $query_str = $query->query_string();

my $request = HTTP::Request->new(GET => $base_url . 'services/data.cgi?' . $query_str);

my $ua = LWP::UserAgent->new( agent => 'oess-app');

#Here is how we authenticate to the entire OEES service...
#just create different request objects and send them through the ua
$ua->credentials( $request->uri()->host_port(),
                    'OEES',
                    'someuser',
                    'somepass'
);

#this actually does the request (in this case its the workgroup request)
my $results = $ua->request($request);

#was the request successful?
if($results->is_success){
    #we need to decode the result into JSON
    #if it succeeded then we have JSON...
    #it might be a good idea to wrap this in an eval just in case
    $results = decode_json( $results->content );
```

```

#now we have a hash representing our JSON object pull out the data we want
$workgroup = $results->{'results'}->[0];

}else{
    #request wasn't successful
    die "Unable to determine workgroup for user";
}

if(!defined($workgroup) || !defined($workgroup->{'workgroup_id'})){
    die "Unable to determine workgroup for user";
}

#now determine what endpoints we want to provision to
#so first we need to request the map
$query = new CGI("action=get_maps");
$query_str = $query->query_string();

$request = HTTP::Request->new(GET => $base_url . 'services/data.cgi?' . $query_str);

my $map_results = $ua->request($request);

if($map_results->is_success){
    $map_results = decode_json($map_results->content);
}else{
    die "Unable to fetch map for network";
}

my $nodes = $map_results->{'results'}->[0]->{'nodes'};

my $endpoint_a;
my $endpoint_z;

#for every node fetch the available interfaces
foreach my $node (keys %{$nodes}){
    $query = new CGI("action=get_node_interfaces&node=$node");
    $query_str = $query->query_string();
    $request = HTTP::Request->new(GET => $base_url . 'services/data.cgi?' . $query_str);
    if(!defined($endpoint_a)){
        $endpoint_a = {node => $node};
    }else{
        if(!defined($endpoint_z)){
            $endpoint_z = {node => $node};
        }
    }
    my $node_interface_results = $ua->request($request);

    if($node_interface_results->is_success){
        $node_interface_results = decode_json($node_interface_results->content);
        $nodes->{$node}->{'interfaces'} = $node_interface_results->{'results'};
    }else{
        die "Unable to get $node interfaces";
    }
}
}

```

```

#now the nodes are fully populated with all of the interfaces we could terminate
#a circuit on

$endpoint_a->{'interface'} = $nodes->{$endpoint_a->{'node'}}->{'interfaces'}->[0];
$endpoint_z->{'interface'} = $nodes->{$endpoint_z->{'node'}}->{'interfaces'}->[0];

$query = new CGI("action=get_shortest_path&node=" . $endpoint_a->{'node'} . "&node=" .
$endpoint_z->{'node'});
$query_str = $query->query_string();
$request = HTTP::Request->new(GET => $base_url . 'services/data.cgi?' . $query_str);
my $shortest_path_result = $ua->request($request);

if($shortest_path_result->is_success){
    $shortest_path_result = decode_json($shortest_path_result->content);
}else{
    die "Unable to get shortest path, are nodes not connected?";
}
print STDERR Dumper($shortest_path_result);
#ok now to determine if vlan 100 is available on both interfaces
$query = new CGI("action=is_vlan_tag_available&node=" . $endpoint_a->{'node'} .
"&interface=" . $endpoint_a->{'interface'} . "&vlan_tag=100");
$query_str = $query->query_string();
$request = HTTP::Request->new(GET => $base_url . 'services/data.cgi?' . $query_str);

my $a_tag_avail_result = $ua->request($request);
if($a_tag_avail_result->is_success){
    $a_tag_avail_result = decode_json( $a_tag_avail_result->content);
}

else{
    die "Unable to determine if tag is available";
}

$query = new CGI("action=is_vlan_tag_available&node=" . $endpoint_z->{'node'} .
"&interface=" . $endpoint_z->{'interface'} . "&vlan_tag=100");
$query_str = $query->query_string();
$request = HTTP::Request->new(GET => $base_url . 'services/data.cgi?' . $query_str);
my $z_tag_avail_result = $ua->request($request);
if($z_tag_avail_result->is_success){
    $z_tag_avail_result= decode_json( $z_tag_avail_result->content);
}

else{
    die "Unable to determine if tag is available";
}

#if our tags are all set then provision our circuit
if($a_tag_avail_result->{'results'}->[0]->{'available'} == 1 && $z_tag_avail_result-
>{'results'}->[0]->{'available'} == 1){

}else{
    #die "Tags were not available"
}

my $shortest_path_links = "";
foreach my $path (@{$shortest_path_result->{'results'}}){
    $shortest_path_links .= "&link=" . $path->{'link'};
}

```

```
$query = new CGI("action=provision_circuit&node=" . $endpoint_a->{'node'} . "&interface=" . $endpoint_a->{'interface'} . "&tags=100" . "&node=" . $endpoint_z->{'node'} . "&interface=" . $endpoint_z->{'interface'} . "&tags=100&workgroup_id=1" . $shortest_path_links . "&provision_time=-1&remove_time=-1&description=test");
$query_str = $query->query_string();
$request = HTTP::Request->new(GET => $base_url . 'services/provisioning.cgi?' . $query_str);
my $provisioning = $ua->request($request);
if($provisioning->is_success){
    $provisioning = decode_json($provisioning->content);
    if($provisioning->{'results'}->{'success'} == 1){
        print "New Circuit created!\n";
    }
}
```